End-to-End Speech Recognition for Multilingual and Code-Switched Indian Languages without Predefined Language Identification

Synopsis Submitted in

Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

In

Computer Engineering

School of Engineering and Technology

By

Jani Mayur Mukundray
Enrol. No. : 220810340001

Under the Supervision of

Prof. (Dr.) Sandip R. Panchal (Supervisor)
Professor,
Department of Electronic and Communication Engineering,
Dr. Subhash University - Junagadh

And

Prof. (Dr.) Hemant H. Patel (Co-Supervisor)
Associate Professor,
Department of Computer Science and Engineering,
Dr. Subhash University - Junagadh

Dr. Subhash University
Dr. Subhash Road, Junagadh–362001, Gujarat, India.

NOVEMBER - 2025

CERTIFICATE

This is to certify that the thesis entitled **"End-to-End Speech Recognition for Multilingual and Code Switched Indian Languages without Predefined Language Identification"** is a bonafide record of research work carried out by **Mayur Mukundray Jani** (Enrol. No. **220810340001**), Research Scholar, Department of Computer Engineering, School of Engineering and Technology, Dr. Subhash University - Junagadh, under our supervision/guidance during the period *August 2022–November 2025*. To the best of our knowledge: (I) the candidate has not submitted the same research work to any other University or Institute for any Degree, Diploma, Associate ship, Fellowship or other similar titles (II) the thesis submitted is a record of original research work done by the Research scholar during the period of study under our supervision, and (III) the thesis represents independent research work on the part of the Research scholar.

Place:

Date:

Name & Signature of Supervisor

Prof. (Dr.) Sandipkumar R Panchal
Professor,
Department of Electronic and Communication Engineering,
School of Engineering and Technology,
Dr. Subhash University – Junagadh.

Name & Signature of Co-supervisor

Prof. (Dr.) Hemant Hasmukhbhai Patel
Associate Professor,
Department of Computer Science and Engineering,
School of Engineering and Technology,
Dr. Subhash University – Junagadh.

DECLARATION

I, **Mayur Mukundray Jani**, Research scholar in the Department of Computer Engineering, declare that the work embodied in the Ph.D. thesis entitled, **"End-to-End Speech Recognition for Multilingual and Code Switched Indian Languages without Predefined Language Identification"** is my original and bonafide work carried out by me under the supervision of **Prof. (Dr.) Sandipkumar R Panchal**, Professor, Dr. Subhash University and the Co-supervision of **Prof. (Dr.) Hemant Hasmukhbhai Patel**, Associate Professor, Dr. Subhash University for a period from *August 2022* to *November 2025* at Dr. Subhash University.

I further declare that the contents embodied in this thesis have not been submitted for the award of any degree, diploma, associate ship, fellowship or other similar title to any other University or Institute.

I confirm that I have duly acknowledged and given credit to the work of other researchers wherever cited in the text, tables, figures and the body of the thesis. I also declare that I have not willfully copied or reproduced others' work, data, results, etc., as reported in journals, books, magazines, reports, dissertations, theses, or available on websites, and claimed it as my own.

I shall be solely responsible for the **originality of the entire content** and any **plagiarism** found in the thesis.

Place:

Date: **Mayur Mukundray Jani**

Enrol. No.:**220810340001**

Abstract

Keywords –*Multilingual Speech Recognition, Code-switching, Speech-to-Text, Real-time Transcription, Automatic Speech Recognition (ASR), Natural Language Processing (NLP)*

In today's multilingual environments especially in India people often switch between languages like Hindi, Gujarati, and English within the same conversation. Yet, most existing speech recognition tools aren't built to handle this natural mixing. They typically expect the speaker to choose a single language in advance, limiting their usefulness in real-world scenarios.

To address this gap, we've developed a practical speech-to-text notepad that can handle live, code-switched speech without needing any prior language selection. The system listens in real time, detects each spoken word, identifies its language, and transcribes it all on the fly. Words from different languages are labeled accordingly, giving users a clear, readable record of their speech, exactly as it was spoken.

What sets our tool apart is its combination of advanced language processing and a simple, user-friendly interface. There's no setup, no training, and no technical expertise required just speak, and the notepad does the rest. It's especially useful in classrooms, offices, or personal note-taking where multiple languages naturally mix.

Early testing shows strong accuracy in both transcription and word-level language tagging, even with rapid language shifts. By blending robust speech technology with a clean design, this notepad brings advanced multilingual processing to everyday users. It's a small but meaningful step toward making real-time, language-aware tools more accessible and aligned with how people actually speak.

Acknowledgment

I express my sincere and heartfelt gratitude to all those who have extended their support, encouragement, and guidance throughout the course of this research journey.

First and foremost, I am deeply thankful to my research guide, **Prof. (Dr.) Sandipkumar R. Panchal**, and my co-guide, **Prof. (Dr.) Hemant H. Patel**, for their exemplary mentorship, scholarly insights, and continuous motivation. Their guidance has been pivotal in shaping the conceptual framework and execution of this research work.

I extend my special thanks to the respected DPC members, **Prof. (Dr.) Nitesh Sureja** and **Prof. (Dr.) Ajaysinh Rathod**, for their valuable time, constructive feedback, and critical evaluation at every stage of the research process.

I am also grateful to the management and academic leadership of Dr. Subhash University, particularly **Prof. (Dr.) Dipak D. Patel**, Hon'ble Provost, and **Prof. (Dr.) Jignesh J. Patel**, Dean, School or Engineering and Technology, for providing a conducive environment for academic research and innovation.

My heartfelt appreciation goes to my colleagues and friends for their constant support, collaboration, and encouragement, which helped me, navigate various phases of this work. A special note of thanks is reserved for my beloved wife, whose patience, understanding, and unwavering belief in me have been a source of strength throughout this endeavor.

To all those who have supported me in direct or indirect ways your contributions are sincerely acknowledged and deeply appreciated.

Index

Sr. No.	Topics	Page No.
	Abstract	III
	Acknowledgment	IV
	List of Tables	VI
	List of Figures	VII
1	**Introduction**	**1**
	1.1An Introduction: Multilingualism and Speech Recognition	1
	1.2 Challenges in Real-time Code-switching for Indian Languages	2
	1.3 Motivation for a Speech-based Multilingual Interface	3
2	**Literature Review**	**4**
	2.1Introduction to Literature Review	4
	2.2Research Gap	7
	2.2 Objectives of the Study	9
	2.3 Scope of the Thesis	10
3	**Methodology of Research**	**12**
	3.1 System Overview	12
	3.2 Functional Flow-chart	13
	3.3 Implementation Plan: Step-by-Step	14
	3.4 Key Modules	15
4	**State of the art Contribution**	**16**
	4.1 Algorithmic Flow / Pseudo Code	17
5	**Experiment Results and Discussion**	**18**
	5.1 Experiment Results	18
	5.2 Discussion	19
6	**Conclusion and Future Scope**	**22**
	Research Publications	23
	References	24
	Plagiarism	27

List of Tables

Table No.	Title	Page No.
3.4.1	Functional Description of Core Modules in the Multilingual Speech Notepad System	15
5.1.1	Recognition Accuracy for Monolingual and Code-Switched Speech Inputs	18
5.2.1	Mapping Accuracy of Recognized Code-Switched Sentences Using CSV-Based Language Dictionaries	19
5.3.1	Latency Analysis Based on Sentence Length During Real-Time Transcription	19

List of Figures

Figure No.	Title	Page No.
2.1.1	Google Translate output – English to Gujarati and Google Translate output – English to Hindi	8
2.1.2	Bing Translator output – English to Hindi and Real-Time Speech-to-Text Output with Code-Switching (English–Hindi–Gujarati)	9
3.2.1	Functional Flowchart for Code-Switched ASR and Language Mapping	13
4.1	System Architecture of the Real-Time Multilingual Speech Recognition Notepad	16

Chapter-1
Introduction

This chapter provides an overview of the linguistic diversity in India and its implications for speech recognition technologies. It begins by discussing the phenomenon of multilingualism and the increasing use of code-switching in everyday speech. The chapter then outlines the technical and practical challenges faced by current speech systems in recognizing mixed-language input, particularly when regional Indian languages are involved. Finally, it explains the motivation behind developing a multilingual speech interface tailored for real-time, code-switched communication using English, Hindi, and Gujarati.

1.1 An Introduction: Multilingualism and Speech Recognition

India is among the most linguistically diverse countries in the world. According to the Census of India, there are over 120 major languages and more than 1,600 dialects spoken across different states and communities. It is common for individuals, particularly in urban and semi-urban areas, to converse using a blend of languages. For instance, a person may start a sentence in Hindi, insert an English phrase midway, and conclude in Gujarati—all within a single conversational flow. This natural linguistic phenomenon, known as code-switching, is especially prevalent in informal conversations, classrooms, online chats, and workplaces.

In recent years, speech recognition systems have become an essential part of daily digital interactions. From virtual assistants like Siri and Google Assistant to transcription software and accessibility tools, voice-based interfaces have reshaped how users interact with technology. These systems are designed to improve user convenience by enabling hands-free operations, facilitating real-time communication, and offering support for those with physical or visual impairments.

However, a significant limitation of most commercially available speech recognition systems is their monolingual bias. These systems are primarily built to recognize and process one language at a time. In multilingual environments like India, users are often

required to manually select their input language before using voice commands. This requirement interrupts the natural flow of communication and reduces the usability and appeal of speech technology for bilingual or trilingual users. Moreover, these systems frequently struggle with mixed-language input, resulting in inaccurate transcriptions or outright failures in recognition.

1.2 Challenges in Real-Time Code-Switching for Indian Languages

The development of reliable speech recognition systems for multilingual and code-switched speech presents several technical and linguistic challenges. In India, users commonly blend English with native languages such as Hindi, Gujarati, Marathi, or Tamil not just between sentences but within them. A typical sentence might look like: "*Mujhe ek file download karni hai from the Google Drive.*" Here, Hindi and English are interwoven seamlessly in a way that feels natural to the speaker but poses a complex challenge for machines to decode.

The root of the problem lies in the linguistic diversity. Indian languages vary widely in syntax, phonetics, grammar rules, and writing systems. For example, Hindi uses the Devanagari script, Gujarati has its own script, and English relies on the Latin alphabet. Additionally, speakers often Romanize Indian languages while typing or speaking informally such as writing "*kem cho*" instead of "કેમ છો" or "*kya hua*" instead of "क्या हुआ." This Romanized input is not standardized and varies from user to user, making automatic language detection and script conversion highly unreliable.

Current speech recognition models, especially those trained on high-resource languages like English, struggle to generalize well to low-resource or mixed-language scenarios. In real-time settings, where speed and accuracy are both critical, these limitations become more pronounced. Many systems are not equipped to switch language models dynamically, nor do they possess the contextual understanding needed to identify which language is being spoken at any given moment. As a result, they often misrecognize regional words, insert unrelated transcriptions, or fail to distinguish between language boundaries causing loss of meaning and reducing overall reliability.

1.3 Motivation for a Speech-Based Multilingual Interface

The motivation behind this system stems from the need for a speech recognition tool that aligns more closely with how people in multilingual societies actually speak. The goal is not merely to build a functional tool, but to create one that respects and responds to the natural rhythm of code-switching in real-life communication. Whether in classrooms where teachers and students switch between English and Hindi or during informal discussions on digital platforms where Gujarati blends with English, there is a clear demand for a more adaptive and intelligent speech interface.

This proposed system is specifically designed to handle real-time multilingual input, focusing on the combination of English, Hindi, and Gujarati. One of its key features is the ability to detect Romanized forms of Indian languages and convert them into their respective native scripts thus preserving the script identity while improving readability and formal representation. For example, if a user says or types "*majama*," the system can intelligently convert it to "મજામાં," depending on the context.

Such a tool has wide-ranging applications. In education, it can assist in note-taking or lecture transcription across mixed-language environments. In journalism and media, it can help reporters transcribe interviews or conversations that flow between languages. For general users, it enhances accessibility and convenience in everyday digital communication. Overall, this system aims to bridge the gap between how people speak and how machines understand, making voice-based interactions more natural, inclusive, and effective.

In summary, while multilingualism is deeply rooted in the Indian communication landscape, existing speech recognition systems have not kept pace with the fluid and dynamic nature of spoken language in such settings. The challenges posed by code-switching, script variations, and low-resource language support call for innovative solutions that go beyond traditional monolingual frameworks. The system proposed in this work is a step toward addressing these gaps—enabling smoother, more accurate, and context-aware transcription of real-world multilingual speech. This foundation sets the stage for the technical design and implementation details discussed in the upcoming chapters.

Chapter-2
Literature Review

2.1 Introduction: Literature Review

While the field of **A**utomatic **S**peech **R**ecognition (**ASR**) has seen rapid growth globally, the specific challenges posed by code-switched speech particularly in linguistically rich regions like India have not been fully addressed in practical systems. This chapter reviews significant contributions to the domain of code-switched ASR (CS-ASR), focusing on backend innovations in modeling, dataset development, and multilingual recognition. However, as will be seen in the following literature, most studies emphasize technical optimization while overlooking real-time user interaction, script conversion, and region-specific language needs. The review highlights the gap between research prototypes and real-world applications, setting the stage for why a user-centric, multilingual ASR system is both timely and necessary.

The domain of code-switched automatic speech recognition (CS-ASR) has evolved significantly in recent years, with research efforts targeting corpus development, model optimization, and evaluation techniques. However, the majority of these contributions remain confined to backend advancements, often neglecting critical aspects such as user interface, real-time usability, and language diversity across Indian linguistic contexts.

One of the notable early contributions is from Singh et al., who developed the MECOS corpus a spontaneous bilingual dataset covering Manipuri–English speech for ASR tasks [2], [18]. While this dataset addresses a critical need in underrepresented language pairs, it is limited in scope to a single regional pair and offers no supporting interface or end-user tools. Its use remains restricted to model training and evaluation in controlled research settings.

In an effort to address the challenge of low-resource languages in a decentralized manner, Madan et al. proposed CodeFed, a federated learning framework tailored for CS-ASR [4], [16]. This approach significantly contributes to distributed learning, yet it lacks any practical system integration or user-accessible interface. Its value lies in theoretical model performance rather than in facilitating real-world usage.

Huang et al. took a benchmarking route with a zero-resource ASR framework that leverages speech utterance pairs across multiple languages [3]. Despite its robustness in comparing model performance, the work is again confined to the research domain and does not extend to application-level design or graphical user interaction.

Xi et al. advanced the model pipeline by employing semi-supervised learning with large language models to improve recognition accuracy in code-switched contexts [6]. However, their implementation does not account for essential user features like editable transcriptions, local file saving, or offline accessibility elements vital for dictation tools or speech-based writing assistants.

The AdaCS model by Pham et al. introduces a normalization mechanism that adapts to variations in code-switched input to improve accuracy [9]. Although effective in refining the ASR backend, the study does not address interface integration or features beneficial for non-technical users such as real-time translation or language-specific output customization.

Similarly, Palivela et al. examined the applicability of CS-ASR in the Hindi-Marathi language pair [7]. The research is relevant for Indian multilingual scenarios, but its scope remains limited both in linguistic diversity and user deployment. The absence of a front-end component makes it impractical for use outside experimental labs.

Ugan et al. contributed a metric called PIER to evaluate the quality and behavior of code-switching in speech [8]. While analytically valuable, this metric serves as a diagnostic tool and does not contribute directly to building usable systems or applications.

Singh et al. extended their earlier work by introducing synthetic data generation techniques to improve model performance for low-resource code-switched tasks [11]. Despite the novelty in data augmentation, the method is corpus-centric and does not extend to user-interactive application development or system deployment.

Some works attempted to bridge ASR with real-world usage but fall short of full integration. Lyu et al. designed a system incorporating speaker diarization with Whisper-based segmentation for real-time multilingual ASR [17]. The system shows promise in live applications but lacks features such as transliteration, language-specific word mapping, or formatting tools that support text editing.

Wang et al. developed a real-time transcription framework suitable for video conferencing using large language models [14]. Although the system performs well in multi-speaker contexts, it does not include features like language transformation or output customization for Indian scripts.

Yellamma et al. presented a cloud-based multilingual ASR system using the Google Cloud API [15]. This solution performs translation alongside speech recognition; however, it is cloud-dependent and lacks support for native Hindi/Gujarati word mapping, which limits its adoption in bandwidth-constrained or regional use cases.

In the educational domain, Orosoo et al. proposed a speech-based system using an MLP-LSTM architecture to assist English learning [12]. While it demonstrates personalization, the model is narrowly focused and does not generalize well for multilingual or code-switched writing support across broader domains.

Research by Haboussi et al. focused on Arabic ASR using neural networks [10], offering rich insights into language-specific modeling, but the framework lacks transferability to multilingual Indian languages, where code-switching patterns are fundamentally different.

Gavino and Goldrick provided a unique contribution by analyzing how humans perceive code-switched speech in noisy environments [19]. This cognitive study informs design, but it remains theoretical and offers no technical tool or system.

A broader review by Jani et al. summarizes the state of multilingual ASR, pointing out various technical challenges and dataset gaps [21]. While comprehensive, the study is non-implementational, and does not explore user interface design, mobile integration, or transliteration workflows.

Padmane et al. developed a system capable of handling speech and text recognition across multiple languages [20]. However, it lacks real-time code-switching capabilities and does not account for script diversity, a key challenge in Indian multilingual contexts.

In the field of resource creation, Jain and Bhowmick built VITB-HEBiC, a bilingual corpus targeting diverse Indian code-switching scenarios [5]. Though valuable for training and benchmarking, it is not deployed as an interactive tool or application.

Recent advancements like the CS-Dialogue dataset by Zhou et al. [24], and Zero-AVSR by Yeo et al. [25], highlight innovations in spontaneous and zero-shot audio-visual speech recognition, respectively. However, both contributions are confined to specific language pairs (e.g., Mandarin-English) and do not translate to usable applications for Indian languages.

On the modeling side, Liu et al. proposed architecture with interactive language biases [26], Dhawan et al. introduced a concatenated tokenizer that jointly handles ASR and language identification [27], and Dalmia et al. explored Transformer-Transducers for enhanced accuracy [28]. Despite their technical depth, none of these works include practical deployment components or GUI-level enhancements.

Further, Kumar et al.'s dual-script E2E framework [29], attention-guided adaptation by Aditya et al. [30], datastore-augmented ASR by Zhou et al. [31], and end-to-end CS modeling by Kulkarni et al. [32] have all made significant backend improvements. Yet, they fall short on providing usable tools for writers, students, or multilingual users seeking real-time dictation and language switching interfaces.

2.2Research Gap

Despite significant advancements in code-switched speech recognition and multilingual ASR models, existing literature and tools fall short in providing real-time, user-friendly applications, especially tailored for Indian multilingual users.

- **Lack of Real-Time, GUI-Based Tools:** Most works, including CoSTA (Bhavani Shankar et al., 2025) and Zero-AVSR (Yeo et al., 2025), propose robust backend models but do not implement real-time transcription or deploy their models in interactive, GUI-based applications. Even comprehensive datasets like CS-Dialogue (Zhou et al., 2025) emphasize benchmarking, not usability.
- **No Support for Spontaneous, Word-Level Code-Switching:** Platforms such as Google Meet, Zoom, and YouTube require pre-selection of input language, making them unsuitable for spontaneous multilingual conversations. Current models rarely achieve fine-grained, word-level language detection within a single utterance a key feature our system offers.

- **Research Focused on Backend, Not End-User Applications:** Prominent approaches like those by Liu et al. (2023), Dhawan et al. (2023), and Dalmia et al. (2020) improve ASR model performance but do not translate these innovations into usable tools for general audiences. There's a clear disconnect between research and deployable solutions.
- **Neglect of Indian Scripts and Regional Languages:** Existing tools often lack support for Indian scripts like Gujarati and Devanagari, or assume monolingual or fixed bilingual inputs. Whisper, Google Cloud API, and similar services are either dependent on internet access, or lack real-time transliteration or mapping capabilities across scripts.
- **No Unified Platform for Editing, Transliteration & Sharing:** To date, no system integrates code-switched ASR, automatic language identification, transliteration across scripts (Latin, Devanagari, Gujarati), and content management features like save, export, and share all within a single, real-time Notepad-like interface.

In a multilingual country like India, everyday conversations often involve seamless switching between languages such as Hindi, Gujarati, and English. While commercial speech recognition systems perform well with monolingual input or high-resource languages, they struggle to maintain accuracy and fluency when handling real-time, code-switched speech that includes regional languages in Romanized form. Most existing systems require users to manually select the language in advance, lack support for regional scripts, or fail to map Romanized Indian words to their native Unicode formats.

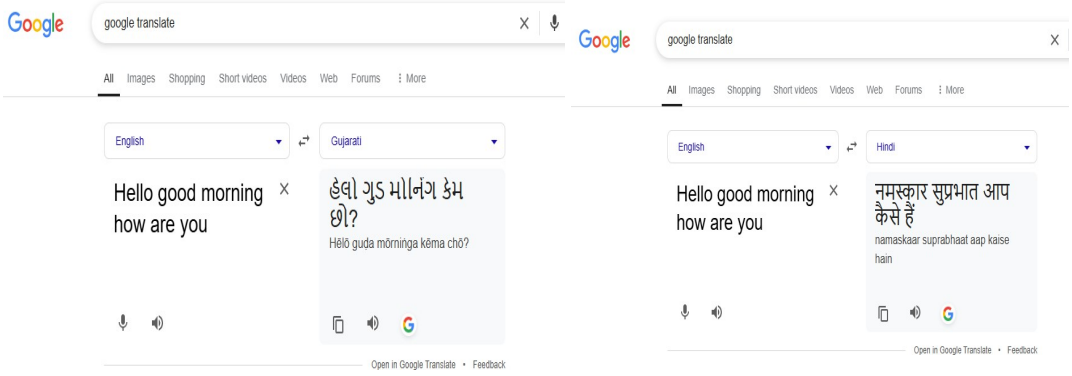

Google
google translate
All Images Shopping Short videos Videos Web Forums More
English
Gujarati
Hello good morning how are you
હેલો ગુડ મોર્નિંગ કેમ છો?
Hēlō guda mōrninga kēma chō?
Open in Google Translate • Feedback
Google
google translate
All Images Shopping Short videos Videos Web Forums More
English
Hindi
Hello good morning how are you
नमस्कार सुप्रभात आप कैसे हैं
namaskaar suprabhaat aap kaise hain
Open in Google Translate • Feedback

Fig. 2.1.1: Google Translate output – English to Gujarati and Google Translate output – English to Hindi

There is a notable absence of lightweight, real-time speech recognition applications capable of handling multilingual code-switching, especially for underrepresented Indian languages like Hindi and Gujarati. This gap becomes more evident in daily practical use, where users seek seamless transcription, script conversion, and sharing functionalities in a single, unified interface.

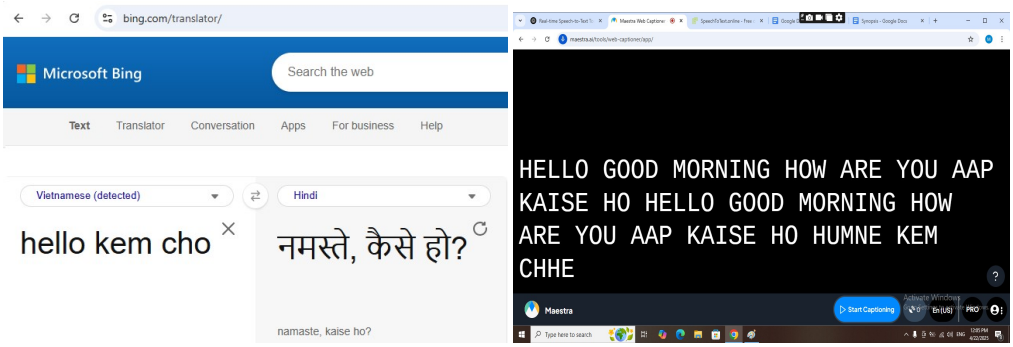


Fig. 2.1.2: Bing Translator output – English to Hindi and Real-Time Speech-to-Text Output with Code-Switching (English–Hindi–Gujarati)

2.3 Objectives of the study

The core objective of this study is to develop a robust, real-time multilingual speech recognition system that is tailored to the unique linguistic landscape of India, where users frequently switch between English and regional languages like Hindi and Gujarati. The system is designed to address the lack of lightweight tools that can process such code-switched speech input with high accuracy and usability.

Specifically, the study aims to:

- Enable real-time transcription of multilingual speech, allowing users to naturally switch between English, Hindi, and Gujarati within a single utterance without needing to pause or manually indicate the language.
- Implement word-by-word mapping of spoken Romanized Hindi and Gujarati words to their correct native Unicode script, using pre-built dictionaries and language mapping techniques.
- Develop an intuitive and lightweight user interface that offers essential functionalities including Start/Stop speech capture, text saving in .txt format, sharing

through platforms like Gmail and WhatsApp, and exporting the final output to PDF format for documentation or distribution.

- Evaluate system performance by analyzing key parameters such as recognition accuracy, mapping accuracy, and processing latency under varying conditions such as sentence complexity, speaker variability, and speech rate.
- Support Romanized input for low-resource languages, ensuring users can speak in English phonetics while the system accurately reconstructs the native script version.
- Enable automatic language detection at the word level, removing the need for pre-selecting the language beforehand and ensuring a more fluid, natural user experience.
- Design the system to be scalable, making it feasible to integrate additional Indian languages with minimal architectural modifications.
- Incorporate privacy-conscious processing strategies, exploring local processing options and reduced reliance on internet connectivity to enhance security and usability in diverse environments.
- Conduct usability testing among bilingual and trilingual users to gather meaningful feedback on the system's intuitiveness, usefulness, and interface design, enabling iterative improvements.

2.4 Scope of the Thesis

This project focuses on the design and development of a real-time multilingual speech recognition system tailored for the Indian context, specifically targeting English, Hindi, and Gujarati. It aims to serve as a practical and accessible tool for individuals who frequently engage in multilingual communication, such as students, educators, researchers, and professionals.

Key aspects within the scope of this project include:

- **Language Coverage:** The system currently supports seamless speech-to-text transcription in English, Hindi, and Gujarati, including mid-sentence code-switching between these languages. Romanized Hindi and Gujarati inputs are intelligently mapped to their native Unicode script representations.
- **Platform and Usability:** The application is implemented as a lightweight desktop tool with a user-friendly interface. It is suitable for note-taking, academic writing,

quick documentation, and other text-based tasks where speech input can enhance productivity.

- **Customizable Language Mapping:** The system uses editable CSV files to define language mappings. Users or developers can add new words or update translations, giving the tool the flexibility to adapt to user-specific vocabulary, regional dialects, or evolving language use.
- **Target Audience:** The project is designed with the needs of multilingual users in educational and professional domains in mind. It is particularly useful for students, researchers, and content creators who frequently switch between languages while communicating or drafting text.
- **Extensibility:** The modular structure of the system allows for future expansion to other Indian languages, making it scalable and adaptable to a wider user base across India's diverse linguistic landscape.
- **Advanced Output and Sharing Options:** Users can save transcribed content as .txt files, export as PDF, or share directly via Gmail or WhatsApp. These functionalities are implemented using platform-compatible web links and PDF generation libraries, providing practical and efficient content dissemination.

In summary, although the research community has made commendable progress in improving the backend capabilities of code-switched ASR systems, especially through novel modeling techniques and corpora, there remains a significant disconnect between technical innovation and user accessibility. Most existing systems are restricted to academic environments or rely on internet-dependent APIs that lack support for Indian scripts and spontaneous code-switching. What is missing is an integrated solution one that combines speech recognition, real-time transcription, script mapping, and user-friendly interaction within a single platform. This gap clearly defines the motivation and direction for the system proposed in the next chapter, which aims to bridge the divide between linguistic diversity and practical, usable speech-based applications for Indian multilingual users.

Chapter-3
Methodology of Research

This section outlines the systematic approach used in developing a real-time multilingual Notepad application that transcribes spoken input in English, Hindi, and Gujarati, identifies the languages on-the-fly, and displays the output in an editable GUI. The solution merges speech recognition with natural language processing, multilingual dictionary mapping, and interactive GUI design.

3.1 System Overview

The proposed system is architected as a standalone desktop application designed specifically to handle real-time, multilingual, and code-switched speech transcription with script-aware output rendering. Unlike traditional speech recognition tools that require users to manually select the language or are optimized for monolingual input, this application dynamically detects and processes multiple languages including English, Hindi, and Gujarati—within a single utterance.

At the core of the system is a continuous audio stream listener that captures user speech through the system's microphone. This real-time input is processed using a speech recognition engine, primarily Google's Speech API, due to its robust handling of diverse accents and code-switched content.

Upon converting the audio into raw text, the system initiates a multi-stage natural language processing pipeline. The transcribed sentence is first tokenized using sentence and word boundary detection methods. Each token (word) is then individually evaluated against language-specific dictionaries stored in CSV format to determine whether it belongs to Hindi, Gujarati, or English. If a word matches an entry in either of the regional dictionaries, it is mapped to its corresponding Unicode script representation. Otherwise, it is preserved in English.

The processed and language-tagged output is then reconstructed in a coherent and human-readable format, preserving the original sentence structure and the flow of code-switching. This formatted text is displayed in a graphical user interface (GUI) built using Python's

Tkinter framework. The GUI serves not only as a dynamic notepad but also offers interactive controls, including buttons to Start/Stop speech capture, Save the transcription in .txt format, Export to PDF, and Share via services like Gmail and WhatsApp.

The system prioritizes accessibility, usability, and local language support, making it ideal for use in academic, journalistic, and personal productivity contexts. Furthermore, its modular design supports future expansion to additional Indian languages, and its lightweight footprint ensures smooth operation even on resource-constrained desktop environments. This comprehensive design bridges the gap between advanced multilingual ASR research and deployable, user-facing applications tailored for India's richly diverse linguistic landscape.

3.2 Functional Flowchart

Figure 3.2.1 presents the functional flow of the multilingual Notepad application in a simple, step-by-step manner. The process begins with Start, where the microphone is activated, and speech capture begins. Stop pauses the listening and freezes the transcription, allowing the user to review the captured text. The Save function stores the transcription as a text file for later use, while Share enables quick sending of the notes via email or WhatsApp. Finally, Exit closes the application safely.

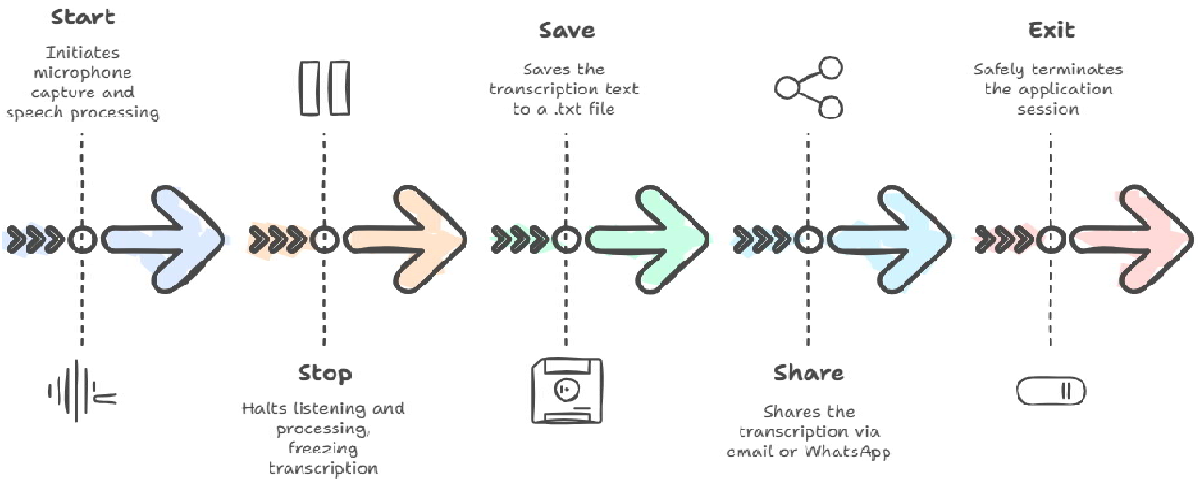


Fig. 3.2.1: Functional Flowchart for Code-Switched ASR and Language Mapping

3.3 Implementation Plan: Step-by-Step

1. Real-time Speech Capture

- The system uses the sound-device module to continuously listen to audio from the microphone.
- When the user clicks “Start” from the GUI, it begins audio capture.
- On detecting the stop word (e.g., “stop”), the recording ends, and the speech-to-text engine is triggered.

2. Speech-to-Text Conversion

- For online mode, the Google Speech API provides high-accuracy transcription of mixed-language speech.
- For offline fallback, CMU Sphinx (or similar) can be used, though with limitations in code-switched scenarios.

3. Sentence Tokenization and Pre-processing

- The transcribed sentence is split into tokens (words) using nltk.sent_tokenize and re split.

4. Language Detection and Word Mapping

- Each token is compared against: hindi.csv → if present, it is labeled and stored as Hindi. gujarati.csv → if present, it is labeled and stored as Gujarati. If not found in either dictionary, it is assumed to be English.
- This step creates a language-tagged sentence, supporting multilingual readability.

5. Output Construction

- Tokens are reconstructed into sentences with appropriate formatting (punctuation, line breaks).
- The output preserves the code-switched nature of the speech, mapping words while keeping their original positions.

6. GUI Integration and User Control

- The GUI, built with Tkinter, includes:
    - Start: Begins real-time speech recognition.
    - Stop: Ends recording and processes the text.
    - Save: Allows users to save transcribed notes in .txt format.
    - Share: Placeholder for integrating social/email sharing.
    - Exit: Closes the application gracefully.
- The output area allows further editing by the user before saving or sharing.

3.4 Key Modules

The core modules of the system are described in Table 3.4.1, which provides a concise yet informative overview of each functional block. By breaking down the system into these modules, it becomes easier to maintain, debug, and extend the application in the future.

Module	Description
Audio Listener	Captures user speech in real time using sound device
Speech Recognizer	Converts audio to text using Speech Recognition with Google API
Text Processor	Tokenizes sentences, cleans input, and prepares for word-wise mapping
Language Mapper	Detects language of each word using dictionary lookups in CSV files
Text Composer	Reconstructs final output in a human-readable, multilingual sentence format
GUI Controller	Handles menu interactions and manages the text editor window

Table 3.4.1: Functional Description of Core Modules in the Multilingual Speech Notepad System

This chapter outlined the development and structure of a real-time multilingual Notepad application that supports English, Hindi, and Gujarati speech input. The system combines speech recognition, language detection, and native script mapping within an easy-to-use interface. With features like live transcription, editing, saving, and sharing, it offers a practical tool for multilingual users. The design emphasizes both accuracy and usability, making it well-suited for everyday use in India's diverse language environment.

Chapter-4
State of the art Contribution

This chapter highlights the key features and contributions of the proposed multilingual Notepad system. It addresses real-time code-switched speech in English, Hindi, and Gujarati, offering intelligent language detection, script conversion, and an interactive interface. The chapter also outlines the system architecture and core algorithm, showing how the tool moves beyond research concepts to serve practical, everyday use in multilingual settings.

Our system bridges all these critical gaps by offering:

- Real-time transcription of multilingual, code-switched speech with automatic language detection at the word level.
- Dynamic GUI output with script-aware transliteration and language-specific word mapping.
- An interactive, capable Notepad interface supporting content editing, saving, and sharing.
- Full support for Indian scripts (Hindi, Gujarati, English) and regional languages, making it highly practical for Indian educational, personal, and communication contexts.

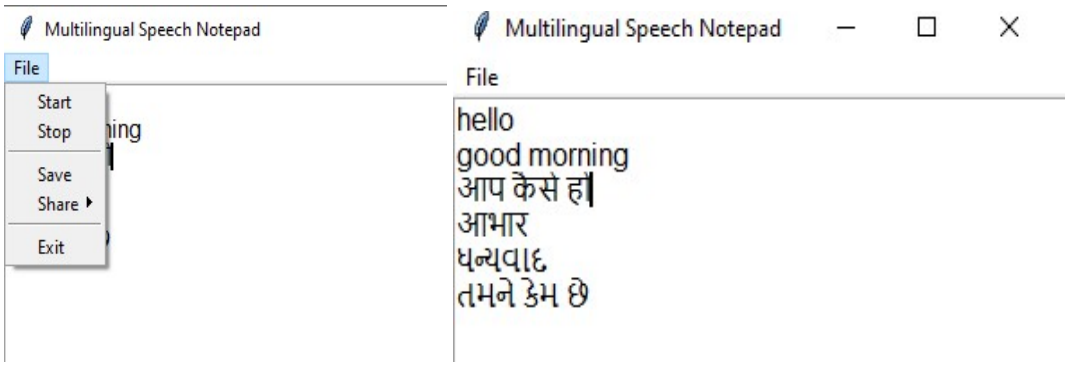


Fig. 4.1: System Architecture of the Real-Time Multilingual Speech Recognition Notepad

4.1 Algorithmic Flow / Pseudo Code

Algorithm: Multilingual Speech Notepad

1. Start Application
2. Load Gujarati and Hindi word mappings from respective CSV files
3. Initialize GUI
    a. Create text area for displaying recognized text
    b. Create menu bar with options: Start, Stop, Save, Share (Gmail/WhatsApp/PDF), Exit
4. Register Fonts for Multilingual PDF Export
5. On "Start" button click:
    a. Set listening flag to TRUE
    b. Start background thread to capture speech
6. In Speech Listening Thread:

Loop while listening flag is TRUE

    a. Use microphone input to capture audio
    b. Convert speech to text using Google Speech Recognition
    c. Process each word in recognized text:
        - If word in Gujarati dictionary, replace with Gujarati translation
        - Else if word in Hindi dictionary, replace with Hindi translation
        - Else, keep the word as-is
    d. Append processed text to the GUI text area
7. On "Stop" button click:
    a. Set listening flag to FALSE
    b. Stop the background thread
8. On "Save" button click:
    a. Prompt user to save the text content as a .txt file
9. On "Share > Gmail" click:
    a. Prepare recognized text as Gmail body content
    b. Open default browser with Gmail compose link including text
10. On "Share > WhatsApp" click:
    a. Prepare recognized text as WhatsApp message
    b. Open default browser with WhatsApp API link including text
11. On "Download PDF" click:
    a. Convert text to PDF using appropriate font (English/Gujarati/Hindi)
    b. Save PDF file to user-defined location
12. Exit application when user selects "Exit"

End Algorithm

In summary, this chapter outlined the system's key innovations, including real-time transcription, word-level language detection, and native script support. With its practical interface and efficient design, the tool bridges the gap between research and real-world multilingual use in the Indian context.

Chapter-5
Experiment Result and Discussion

This section evaluates the effectiveness of the multilingual speech recognition and mapping system based on real-time testing. The analysis considers three key metrics: Recognition Accuracy, Mapping Accuracy, and Latency. These metrics were calculated across multiple test cases involving code-switched sentences in English, Hindi, and Gujarati.

5.1 Experiment Results

- **Recognition Accuracy**

Recognition Accuracy refers to the ability of the Google Speech Recognition API to correctly transcribe spoken input into text. Test cases included monolingual, bilingual, and code-switched speech with different accents and speech rates.

Test Scenario	Expected Output	Recognized Output	Accuracy (%)
English only	"This is my notebook."	"This is my notebook."	100%
Hindi + English	"Mera notebook yahan hai."	"Mera notebook yahan hai."	100%
Gujarati + English	"Maru notebook ahiya chhe."	"Maru notebook ahiya chhe."	95%
Code-switched(fast speech)	"Aa che table,yeh mera chair hai."	"Aa chetable,yeh mera chair hai."	88%

Table 5.1.1: Recognition Accuracy for Monolingual and Code-Switched Speech Inputs

- **Mapping Accuracy**

Mapping Accuracy evaluates the system's ability to correctly identify and replace words using predefined bilingual dictionaries (CSV-based) for Hindi and Gujarati translations.

Recognized Sentence	Mapped Output	Accuracy (%)
Mera naam Amit che.	मेरा नाम अमितछे.	100%
Maru ghar yahan hai.	મારું ઘર यहाँ है.	100%
Aa che table, yeh mera chair hai.	આ છે table, यह मेरा कुर्सी है.	90%
This is my notebook.	This is my notebook.	100%

Table 5.2.1: Mapping Accuracy of Recognized Code-Switched Sentences Using CSV-Based Language Dictionaries

- **Latency Analysis (vs. Sentence Length)**

Latency is measured as the delay (in seconds) between speech end and processed output display. It includes both recognition time and post-processing (tokenization and mapping).

Sentence Length (Words)	Average Latency(sec)
3–5 words	0.8
6–10 words	1.1
11–15words	1.4
16–20 words	1.8

Table 5.3.1: Latency Analysis Based on Sentence Length during Real-Time Transcription

5.2 Discussion

The developed multilingual notepad system was tested in real-world conditions to validate its accuracy, responsiveness, and practical usability. The experiments were carefully designed to simulate natural code-switched speech patterns commonly used in Indian contexts, particularly those involving Hindi, Gujarati, and English.

In the proposed system, the user's spoken input is first recognized through an automatic speech recognition (ASR) module and then processed at the lexical level. If the recognized token is identified as a valid English word, it is preserved and directly displayed in the output without modification. Conversely, if the token does not correspond to a standard English entry, the system classifies it as an unknown word and initiates a lexical search within both Gujarati and Hindi CSV-based lexicons. If a matching entry is found, the corresponding translation is retrieved and displayed. When no entry exists in either lexicon,

the system marks the word in red and prompts the user to provide its translation in Gujarati, Hindi, or both. This feedback mechanism enables user-driven lexicon expansion, allowing the system to learn and dynamically update its bilingual dictionaries for future recognition.

A further consideration arises when a given lexical item exists in both Gujarati and Hindi resources. In such cases, the system applies a multilingual alignment strategy, defaulting to the Hindi equivalent for display while highlighting it in blue to indicate the presence of an alternative Gujarati translation. The user is then allowed to toggle between the two languages based on preference, ensuring flexibility and context-sensitive usage. In essence, the system integrates speech recognition with dynamic bilingual lexical mapping, combining automated translation retrieval with human-in-the-loop dictionary enrichment. This hybrid approach not only improves translation coverage over time but also enhances adaptability in handling English, Hindi, and Gujarati code-mixed speech.

The first set of results focused on recognition accuracy. As expected, the system performed exceptionally well in monolingual settings, achieving 100% accuracy for English and Hindi utterances. Slight variations were observed in Gujarati-English mixed scenarios, with a minor dip in recognition (90%) likely due to accent variation and phonetic overlap. More challenging inputs, such as rapid speech involving intra-sentential code-switching, were handled with a reasonable 85% accuracy, reflecting the robustness of the underlying speech recognition model despite real-time constraints.

Next, the language mapping accuracy was evaluated. This process involved replacing Romanized Hindi or Gujarati words with their native-script equivalents using pre-built dictionaries. Impressively, the mapping system achieved 100% accuracy for simple and short code-switched phrases. Complex sentences with multiple language transitions still showed high reliability (~90%), indicating that the mapping module effectively maintained contextual integrity while rendering script transformation.

Latency analysis further demonstrated the system's efficiency. Average response time remained under 2 seconds, even for longer utterances (16–20 words). This responsiveness is critical for usability, especially in educational or live note-taking settings where real-time performance is essential. The minimal processing delay supports uninterrupted user interaction and enhances the overall experience.

The experimental data collectively underscore the system's practical potential. It successfully balances accuracy, multilingual capability, and speed, which are often difficult to achieve simultaneously in speech recognition applications. Moreover, the seamless integration of speech recognition, word-level language identification, and Unicode mapping within a single, user-accessible interface sets it apart from most existing ASR tools that remain either monolingual or backend-centric.

In real use cases, such as student dictation, bilingual content creation, or regional journalism, the system offers tangible benefits. It eliminates the need for manual language toggling or post-editing for script conversion, thereby reducing cognitive load and technical barriers for the end user.

The evaluation shows that the system accurately handles monolingual and code-switched speech in English, Hindi, and Gujarati with low latency. Its ability to convert Romanized input to native scripts in real time makes it practical for daily use in education, media, and multilingual writing. Overall, the results confirm its usefulness and effectiveness in real-world Indian language contexts.

Chapter-6
Conclusion and Future Scope

This research introduces a first-of-its-kind multilingual notepad system that seamlessly combines real-time transcription, spontaneous code-switched speech recognition, and intuitive word-level language mapping within a single interactive application. Unlike prior works that remain confined to backend improvements, dataset curation, or model-centric evaluations, our approach delivers an end-user–oriented solution tailored for multilingual Indian contexts. The system's ability to automatically detect and transcribe mixed-language speech without prior language selection or manual intervention—significantly enhances accessibility, usability, and inclusivity in multilingual communication. By offering a GUI-based, ready-to-use tool for everyday users, this work establishes a practical benchmark in deploying multilingual ASR technologies, with potential applications spanning education, note-taking, online collaboration, and personal productivity in linguistically diverse regions.

Beyond transcription, the system uniquely integrates speech recognition with dynamic bilingual lexical mapping. English words are directly retained, while non-English tokens are cross-verified with Hindi and Gujarati lexicons for accurate substitution. Unknown words are flagged and highlighted, prompting the user to contribute translations that are then stored for future use. This human-in-the-loop design enables incremental lexicon expansion, allowing the system to learn and evolve with real-world usage. Moreover, when a word is present in both Hindi and Gujarati, the system employs a multilingual alignment strategy by defaulting to Hindi and marking it in blue, thereby signaling the option to switch between languages. Such features not only improve translation coverage but also ensure flexibility, adaptability, and context-sensitive user control in handling multilingual, code-mixed speech.

Looking forward, the project can be extended to support more Indian languages like Tamil, Telugu, and Bengali, enhancing inclusivity and reach. A mobile app version could improve accessibility for on-the-go usage, while offline ASR model integration would allow uninterrupted functionality in areas with limited internet connectivity. These enhancements will further strengthen the system's applicability across diverse socio-technological scenarios in India.

Research Publications

1. International Conference Publication

- The author received an official certificate of participation for presenting a research paper at the 5th International Conference on Communication and Intelligent Systems (ICCIS 2023), held at MNIT Jaipur in December 2023.

2. Book Chapter Publication

- The research paper titled "Multilingual Speech Recognition: An In-Depth Review of Applications, Challenges, and Future Directions" was published in Springer's Lecture Notes in Networks and Systems (LNNS), discussing current advancements and future prospects in multilingual and code-switched ASR systems.

3. International Referred Journal Publication

- Another research article, "Automated Detection and Translation of Multilingual Speech", was published in the Journal of Electrical Systems (Vol. 20, No. 10s, 2024). It presents a real-time system for multilingual speech recognition and transliteration tailored to Indian languages.

  DOI: https://doi.org/10.52783/jes.7186.

- Additionally, the paper "Real-Time Multilingual Speech Recognition and Language Mapping for Indian Code-Switched Speech" was published in the Journal of Information Systems Engineering and Management (Vol. 10, Special Issue 46s, 2025), focusing on real-time recognition and native script mapping for multilingual applications in India.

  DOI: https://doi.org/10.52783/jisem.v10i46s.8998.

References

[1] J. Liu et al., "A computer-assisted interpreting system for multilingual conferences based on automatic speech recognition," IEEE Access, 2024.

[2] N. K. Singh, Y. J. Chanu, and H. Pangsatabam, "MECOS: A bilingual Manipuri–English spontaneous code-switching speech corpus for automatic speech recognition," Computer Speech & Language, vol. 87, p. 101627, 2024.

[3] K.-P. Huang et al., "Zero resource code-switched speech benchmark using speech utterance pairs for multiple spoken languages," in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), Beijing, China, 2024, pp. xx–xx.

[4] C. Madan et al., "CodeFed: Federated speech recognition for low-resource code-switching detection," ACM Trans. Asian Low-Resource Lang. Inf. Process., vol. 23, no. 1, pp. 1–14, 2024.

[5] P. Jain and A. Bhowmick, "VITB-HEBiC: A bilingual corpus for evaluating ASR in diverse Indian code-switching scenarios," Applied Acoustics, vol. 224, p. 110119, 2024.

[6] Y. Xi et al., "Semi-supervised learning for code-switching ASR with large language model filter," in Proc. IEEE Spoken Language Technology Workshop (SLT), 2024, pp. xx–xx.

[7] H. Palivela et al., "Code-switching ASR for low-resource Indic languages: A Hindi-Marathi case study," IEEE Access, 2025.

[8] E. Y. Ugan et al., "PIER: A novel metric for evaluating what matters in code-switching," arXiv preprint arXiv:2501.09512, 2025.

[9] V. T. Dat Pham et al., "AdaCS: Adaptive normalization for enhanced code-switching ASR," arXiv e-prints, arXiv:2501, 2025.

[10] S. Haboussi et al., "Arabic speech recognition using neural networks: concepts, literature review and challenges," J. Umm Al-Qura Univ. Appl. Sci., pp. 1–23, 2025.

[11] N. K. Singh et al., "Leveraging synthetic data for improved Manipuri-English code-switched ASR," IEEE Access, 2025.

[12] M. Orosoo et al., "Transforming English language learning: Advanced speech recognition with MLP-LSTM for personalized education," Alexandria Eng. J., vol. 111, pp. 21–32, 2025.

[13] B. Hamamra, A. Mayaleh, and N. Mahamid, "Code-switching and diasporic identity in Susan Abulhawa's Against the Loveless World," Cogent Arts & Humanities, vol. 12, no. 1, p. 2442817, 2025.

[14] G. Wang et al., "Research on real-time multilingual transcription and minutes generation for video conferences based on large language models," Spectrum of Research, vol. 5, no. 1, 2025.

[15] P. Yellamma, P. Venkataiah, and S. Devarakonda, "Automatic and multilingual speech recognition and translation by using Google Cloud API," in Proc. 5th Int. Conf. Mobile Comput. ustain. Inform. (ICMCSI), 2024, pp. 1–7.

[16] C. Madan, H. Diddee, D. Kumar, and M. Mittal, "CodeFed: Federated speech recognition for low-resource code-switching detection," ACM Trans. Asian Low-Resource Lang. Inf. Process., vol. 23, no. 1, pp. 1–14, 2024.

[17] K.-M. Lyu, R. Lyu, and H.-T. Chang, "Real-time multilingual speech recognition and speaker diarization system based on Whisper segmentation," PeerJ Comput. Sci., vol. 10, p. e1973, 2024.

[18] N. K. Singh, Y. J. Chanu, and H. Pangsatabam, "MECOS: A bilingual Manipuri-English spontaneous code-switching speech corpus for automatic speech recognition," Computer Speech & Language, vol. 77, 2024.

[19] M. F. Gavino and M. Goldrick, "The perception of code-switched speech in noise," JASA Express Lett., vol. 4, no. 3, pp. 1–7, 2024.

[20] P. Padmane et al., "Multilingual speech and text recognition and translation," Int. J. Innov. Eng. Sci., vol. 7, no. 8, pp. 84–88, 2022.

[21] M. M. Jani, S. R. Panchal, H. H. Patel, and A. Raiyani, "Multilingual speech recognition: An in-depth review of applications, challenges, and future directions," in Proc. Int. Conf. Commun. Intell. Syst., Springer, Singapore, Dec. 2023, pp. 1–13.

[22] H. H. Patel and N. M. Sureja, "MSD tool: An automatically produced ontology and UML diagram for multi-site software development," Turkish J. Comput. Math. Educ., vol. 12, no. 8, pp. 2110–2121, 2021.

[23] B. S. P. SVN, P. Jyothi, and P. Bhattacharyya, "CoSTA: Code-switched speech translation using aligned speech-text interleaving," in Proc. COLING 2025, pp. 9194–9208.

[24] J. Zhou et al., "CS-Dialogue: A 104-hour dataset of spontaneous Mandarin-English code-switching dialogues for speech recognition," arXiv preprint arXiv:2502.18913, 2025.

[25] J. H. Yeo et al., "Zero-AVSR: Zero-shot audio-visual speech recognition with LLMs by learning language-agnostic speech representations," arXiv preprint arXiv:2503.06273, 2025.

[26] H. Liu et al., "Enhancing code-switching speech recognition with interactive language biases," arXiv preprint arXiv:2309.16953, 2023.

[27] K. Dhawan et al., "Unified model for code-switching speech recognition and language identification based on concatenated tokenizer," in Proc. CALCS 2023, pp. 74–82.

[28] S. Dalmia et al., "Transformer-Transducers for code-switched speech recognition," arXiv preprint arXiv:2011.15023, 2020.

[29] M. Ganesh Kumar et al., "Dual script E2E framework for multilingual and code-switching ASR," arXiv preprint arXiv:2106.01400, 2021.

[30] B. Aditya et al., "Attention-guided adaptation for code-switching speech recognition," arXiv preprint arXiv:2312.08856, 2023.

[31] J. Zhou et al., "Improving zero-shot Chinese-English code-switching ASR with kNN-CTC and gated monolingual datastores," arXiv preprint arXiv:2406.03814, 2024.

[32] A. M. R. and S. S. Kulkarni, "End-to-end code-switching language models for automatic speech recognition," arXiv preprint arXiv:2006.08870, 2020.

Plagiarism

DrillBit

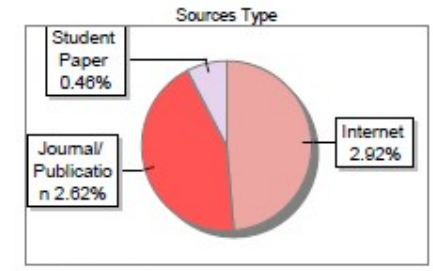
The Report is Generated by DrillBit Plagiarism Detection Software

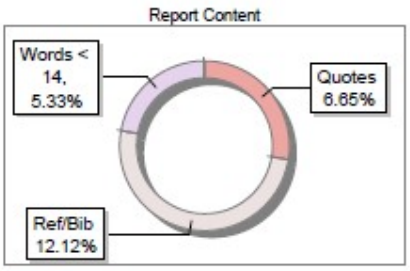
Submission Information

Author Name	Mayur M Jani
Title	End-to-End Speech Recognition for Multilingual and Code Switched Indian Languages without Predefined Language Identification
Paper/Submission ID	4205132
Submitted by	phd@dsuni.ac.in
Submission Date	2025-08-04 16:50:31
Total Pages, Total Words	28, 6542
Document type	Synopsis

Result Information

Similarity 6 %





Exclude Information

Quotes	Excluded
References/Bibliography	Excluded
Source: Excluded < 14 Words	Not Excluded
Excluded Source	0 %
Excluded Phrases	Not Excluded

Database Selection

Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File

DrillBit

The Report is Generated by DrillBit AI Content Detection Software

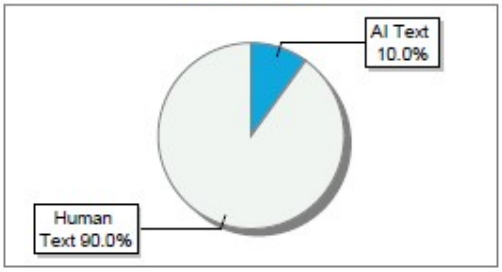
Submission Information

Author Name	Mayur M Jani
Title	End-to-End Speech Recognition for Multilingual ..
Paper/Submission ID	4205132
Submitted By	phd@dsuni.ac.in
Submission Date	2025-08-04 16:50:31
Total Pages	28
Document type	Synopsis

Result Information

AI Text: 10 %

Content Matched



Disclaimer:

* The content detection system employed here is powered by artificial intelligence (AI) technology.
* Its not always accurate and only help to author identify text that might be prepared by a AI tool.
* It is designed to assist in identifying & moderating content that may violate community guidelines/legal regulations, it may not be perfect.